Full Stack Development with MERN

Project Documentation

1. Introduction

- **Project Title:** FreelanceFinder: Discovering Opportunities, Unlocking Potential
- **Team Members:** Duggirala Venkata Surya – Backend development
  Byreddy Vijaya Gopala Srinadh – Database development
  Karri Shanmukhi Saisridevi – Frounted Development
  Konakalla Anjali – Project Imlementatin and Project setup

2. Project Overview

- **Purpose:** The purpose of the SB Works platform is to bridge the gap between clients seeking specialized services and freelancers offering diverse skillsets. It aims to create an efficient, transparent, and secure freelancing environment where clients can post projects and freelancers can bid on and complete them with ease. SB Works empowers individuals like Sarah, a recent graduate, to build a strong freelance portfolio and connect with clients through a seamless and supportive platform.
- **Features:**

1.User-Friendly Interface: Intuitive and responsive design for easy navigation by clients, freelancers, and admins.
Project Posting & Bidding: Clients can post projects while freelancers can bid based on their skills and experience.
2.Integrated Chat System: Enables real-time communication and collaboration between clients and freelancers.
3. Secure File Submissions: Freelancers can submit completed work securely through the platform.
4.Admin Oversight: Ensures secure transactions and resolves disputes while maintaining platform integrity.
5.Feedback System: Clients can review submitted work and provide ratings and comments.
6.Real-Time Notifications: Keeps users updated with project activities and new opportunities.
7.Scalable Architecture: Powered by MongoDB, Express.js, and REST APIs with a responsive frontend using Bootstrap and Material UI.

3. Architecture

- **Frontend:** The frontend of SB Works is built using React.js, providing a dynamic, responsive, and modular user interface. It leverages Axios for API integration and React Router for smooth navigation between pages. UI elements are styled with Bootstrap and Material UI to enhance user experience. Components are structured by user roles (Client, Freelancer, Admin) and feature-based directories, ensuring maintainability and scalability.
- **Backend:** The backend is developed using Node.js and the Express.js framework. It exposes a set of RESTful APIs to handle user authentication, project management, messaging, and submission workflows. Middleware ensures request validation, session management, and error handling. The backend also enforces role-based access control for clients, freelancers, and admins to ensure secure and appropriate feature access.

- **Database:** SB Works uses MongoDB for flexible, schema-less data storage, making it ideal for the evolving data needs of a freelancing platform. Key collections include:
  - Users: Stores freelancer, client, and admin profiles with roles and credentials.
  - Projects: Contains project details, bidding data, and project status.
  - Bids: Tracks freelancer proposals, bid amounts, and timelines.
  - Messages: Maintains communication threads between users.
  - Submissions: Handles file uploads and final work delivery**.**
-

4. Setup Instructions

- **Prerequisites:** To run the SB Works freelancing platform locally, ensure you have the following installed:

  - Node.js (version 14.x or above)
  - npm (Node Package Manager)
  - MongoDB (either local installation or MongoDB Atlas account)
  - Git (to clone the repository)

5. Folder Structure

- **Client:** The frontend is developed using React and organized into role-based and reusable components for maintainability and scalability.

```
client/
│
├── public/              # Public assets like index.html and favicon
├── src/
│   ├── components/      # Reusable UI components (Navbar, Login, Register, etc.)
│   ├── context/         # React Context for global state management
│   ├── pages/
│   │   ├── admin/       # Admin dashboard and management pages
│   │   ├── client/      # Client-specific pages (Post project, My Projects)
│   │   ├── freelancer/  # Freelancer-specific pages (Bid, Submit work)
│   ├── App.js           # Main app component with routes
│   ├── index.js         # Entry point
│   └── utils/           # Utility functions (API helpers, validators)
└── package.json         # Project metadata and dependencies
```

6. Server:

```
server/
│
├── controllers/          # Logic for handling requests (auth, projects, users)
├── models/               # Mongoose schemas for Users, Projects, Bids, Messages
├── routes/               # Express routes grouped by modules (auth, users, projects)
├── middleware/           # Custom middleware (auth check, error handling)
├── config/               # Database config and environment setup
├── uploads/              # Folder to store submitted files
├── .env                  # Environment variables (not committed)
├── server.js             # Entry point for Express server
└── package.json          # Backend dependencies and scripts
```

7. Running the Application:

To run the SB Works platform locally after completing setup and installation, follow these steps in two separate terminal windows or tabs:

- **Frontend:** Navigate to the client directory
  cd client
  npm start
- **Backend:** `npm start` in the server directory.
  cd server
  npm start

8. API Documentation

- Authentication POST /api/auth/register – Register a user Body: username, email, password, role Response: token, user
- POST /api/auth/login – Login Body: email, password Response: token, user
- 📁 Projects POST /api/projects – Create a project (Client only) Body: title, description, budget Auth: Required
- GET /api/projects – Get all projects Response: List of projects
- 📨 Bids POST /api/bids – Submit bid (Freelancer) Body: projectId, amount, message Auth: Required
- 👤 Users GET /api/users/:id – Get user profile Response: username, role, etc.
- 📩 Submissions POST /api/submissions – Submit project files FormData: projectId, file Auth: Required

9. Authentication

SB Works uses **JWT (JSON Web Tokens)** for authentication and role-based authorization:

- **Login Process:**
  - Users log in via email and password.
  - On success, the server issues a **JWT token**.
  - Token is stored in the browser's **localStorage**.
- **Token Usage:**
  - Token is included in the Authorization header for all protected API requests.
  - Example: Authorization: Bearer <token>

10. User Interface

- SB Works offers a clean, responsive UI designed with React, Bootstrap, and Material UI:
- Key UI Features: Landing Page – Overview of platform and call to action.
- Client Dashboard – Project creation and management.
- Freelancer Dashboard – Browse and bid on projects.
- Admin Panel – User and project oversight.
- Chat Interface – In-platform communication between clients and freelancers.
- Project Submission & Review – Upload and feedback system.

11. Testing

- Testing Strategy: Manual Testing:
- Performed on all major user flows including:
- User registration and login
- Project posting and bidding
- File submission and feedback
- Role-based dashboard access
- API Testing:
- Done using Postman to verify backend endpoints.
- Covered:
- Auth routes (login/register)
- Project creation and retrieval
- Bid submission and user profiles
- Unit Testing (Planned for future):
- Framework: Jest (for backend logic)
- Focus: Controllers and middleware functions

- **Tools Used:** Postman – API testing and response validation
  - Chrome DevTools – UI inspection and debugging
  - MongoDB Compass – Database verification

12. Demo Video Link

https://drive.google.com/file/d/19drB01P5vrVDmYa4v2RiWBVpd4IWKUAb/view?usp=sharing

13. Known Issues

- 🔄 Real-Time Messaging: Currently, the chat system does not support real-time updates (no WebSocket/socket.io integration).
- 🧪 Validation: Some form fields lack comprehensive client-side validation (e.g., file type, input length).
- 📩 Notification System: Email or in-app notifications are not yet implemented.
- ⚠️ Error Handling: Backend error messages need more descriptive responses for debugging.

14. Future Enhancements

- 💬 Real-Time Chat: Integrate Socket.io for live messaging and notifications.
- 💳 Payment Gateway Integration: Add Stripe/Razorpay for secure client-freelancer transactions.
- 📊 Admin Analytics Dashboard: Visual insights into platform usage, top freelancers, active projects.

- 🏅 Freelancer Endorsements: Enable rating, reviews, and skill endorsements from clients.
- 📱 Mobile App: Build a cross-platform mobile app using React Native.
- 🔐 Two-Factor Authentication (2FA): Improve login security with OTP/email verification.
- 🌐 Multi-language Support: Localize the app for broader user reach.